CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

Scheduling In-House Transport Vehicles to Feed Parts to Automotive Assembly Lines

Simon Emde
Michel Gendreau

October 2015

CIRRELT-2015-50

Bureaux de Montréal :
Université de Montréal
Pavillon André-Aisenstadt
C.P. 6128, succursale Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Université Laval · McGill · Concordia · ÉTS · UQÀM · HEC Montréal · Polytechnique Montréal · Université de Montréal

Scheduling In-House Transport Vehicles to Feed Parts to Automotive Assembly Lines

Simon Emde[1,*], Michel Gendreau[2]

[1] Friedrich-Schiller-Universität Jena, Lehrstuhl für Operations Management, Carl-Zeiß-Straße 3, D-07743 Jena, Germany

[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Mathematics and Industrial Engineering, Polytechnique Montréal, P.O. Box 6079, Station Centre-ville, Montréal, Canada H3C 3A7

Abstract. Due to exorbitant product variety, very limited space, and other factors, organizing efficient and timely deliveries of parts and subassemblies to final assembly within the factory is one of the most pressing problems of modern mixed-model assembly production. Many automobile producers have implemented the so-called "supermarket" concept to transfer material to the assembly line frequently and in small lots. Supermarkets are decentralized logistics areas on the shop floor where parts are intermediately stored for nearby assembly cells, to be ferried there by small transport vehicles (called tow trains or tuggers). This paper tackles the operational problem of drawing up schedules for these tow trains, such that the assembly line never starves for parts while also minimizing in-process inventory, thus satisfying just-in-time goals. We present complexity results as well as exact and heuristic solution methods. In a computational study, the procedures are shown to perform very well, solving realistic instances to (near-) optimality in a matter of minutes. We also provide some managerial insight into the right degree of automation for such a part feeding system.

Keywords: Mixed-model assembly lines, just-in-time, production logistics, tow trains.

Acknowledgements. This work was supported by a fellowship within the Postdoc-Program of the German Academic Exchange Service (DAAD).

Results and views expressed in this publication are the sole responsibility of the authors and do not necessarily reflect those of CIRRELT.

Les résultats et opinions contenus dans cette publication ne reflètent pas nécessairement la position du CIRRELT et n'engagent pas sa responsabilité.

---

* Corresponding author: simon.emde@uni-jena.de

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2015

© Emde, Gendreau and CIRRELT, 2015

1 Introduction

In recent decades, logistics problems have increasingly become the center of attention for many companies in the automotive sector. This is mostly due to the exorbitant product variety most automobile producers today offer, aiming to fulfill even niche customer demands. Owing to individually (de-)selectable options (e.g., leather trim or car radio), the number of theoretically available model variants is in the trillions (Boysen et al., 2009). Seeing that the production rate of a typical automotive assembly line can easily approach 1 car per minute, the enormous amount and diversity of parts required in final assembly becomes obvious. It is practically impossible to keep all those parts in stock at the factory, let alone at the assembly workstations themselves, for any prolonged amount of time. Consequently, almost all automobile manufacturers implement (or aim to implement) just-in-time logistics at least for a portion of their supplies, such that parts are only brought from one production stage to next when and if required.

This puts a heavy strain on the logistics systems as it necessitates a large number of small-lot deliveries. For example, at the Volkswagen plant in Wolfsburg, up to 750 trucks per day need to be processed (autogramm, 2007); similarly, at the BMW plant in Dingolfing 400 trucks, carrying around 13,000 individual shipments, need to be handled each day (Battini et al., 2013). Just-in-time concerns are especially pressing in final assembly itself, where, on the one hand, space is extremely scarce, prohibiting large safety stocks, and, on the other hand, even slight delays can have very dire consequences. In the worst case, if an important part is missing from the workstation that needs it, the whole assembly line has to be halted, leading to hundreds of workers being idle and lost sales of one car per minute.

In order to keep modern mixed-model assembly systems well supplied, many automobile producers use so-called just-in-time supermarkets, that is, decentralized logistics areas on the shop floor, where parts are intermediately stored to then be brought to the assembly line in small lots. Supermarkets mainly consist of a couple of shelves, set up for ease of access, not necessarily space-efficient storage. Parts are brought there in fairly large lots (e.g., pallets) by industrial trucks, which are then unpacked and presorted. When demand at the workstations within the supermarket's area of responsibility occurs, a bin for that station is prepared. These bins often contain complete kits of parts, prepared for specific models in the exact same order in which the workpieces to be assembled move down the line to reduce the searching effort and unproductive walking times of the assembly worker (Bozer and McGinnis, 1992, Limère et al., 2012). Moreover, these comparatively small, standard-size bins can be stored in easily accessible racks, improving ergonomics (Finnsgård et al., 2011), which is becoming increasingly important for many manufacturing companies due to an ageing workforce in the industrialized countries.

While industrial trucks are used to take parts in large quantities to the supermarket, these vehicles are poorly suited to distributing small bins to individual stations. This work is usually performed by so-called tow trains (or tuggers), made up of an electrically powered towing vehicle connected to a small number of wagons. The tow train is loaded in the supermarket with the bins of parts destined for the stations that lie on its predefined route. It then sets off to visit these stations one after another to deliver the respective

bins. Once it has completed its tour, it will return to the supermarket to be refilled for its next tour. Figure 1 shows the concept of such an in-house logistics system.

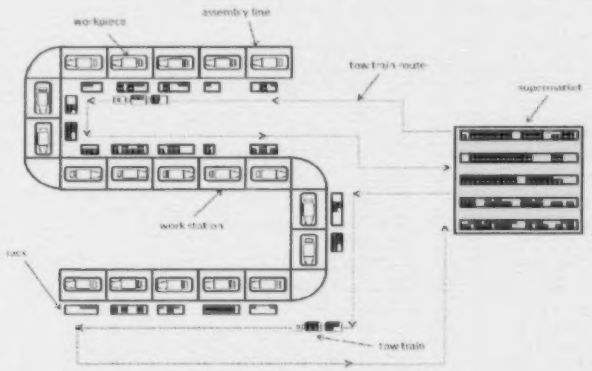


Figure 1: An assembly line fed parts by two tow trains.

Tow trains usually operate on a strict schedule. Due to the given fixed cycle times in the automotive industry and the predetermined production sequences, communicated to suppliers 3 to 4 days in advance before production starts, the exact moment in time (i.e., work cycle) when demand for a specific part at a specific station occurs can be determined well in advance (Emde et al., 2012). Some automobile producers are even experimenting with fully automated delivery systems, where tow trains are operated as driverless AGVs (automated guided vehicles), which can dock at specially designed racks at the stations and unload their cargo in a matter of seconds without the need for any human intervention. In these scenarios, the tow train might as well stop at every station on every tour, because the stopover times may be negligible compared to the driving times. Most systems in use today, however, still have human operators. In some assembly plants, at least the bin exchange at the stations is automated (through the use of so-called shooter racks, Emde et al., 2012), although in other plants, empty and full bins still have to be manually swapped. This may require a more substantial amount of time, making the decision which stations to stop at and which stations to skip on a given tour an integral part of the planning problem.

This paper tackles the optimization problem, occurring on a daily basis in mixed-model assembly plants, of determining the schedule (i.e., how often and when does the tow train leave the supermarket to tour the stations on its given route) and the load (i.e., at which stations does the tow train stop on each tour and how many bins should it unload there) of a tow train responsible for one predetermined route.

The remainder of this paper is structured as follows. In Section 2 we will give an overview over prior work relevant to our problem, Section 3 defines the problem and Section 4 investigates the computational complexity of the main problem as well as of an important special case. In Section 5, we present a MIP model and a heuristic decomposition procedure, which are tested in a comprehensive computational study in

Section 6, where we also offer some decision support on the right degree of automation for a supermarket-based part feeding system. Finally, Section 7 concludes the paper.

2 Literature review

Despite their widespread application in the automotive and similar industries, the scheduling of tow trains has not received very expansive attention in the scientific literature so far. A recent brief survey on supermarket-based part feeding systems is provided by Battini et al. (2013).

Emde et al. (2012) distinguish between four decision problems related to organizing the part feeding process with tow trains: First, location planning for the supermarkets on the shop floor; second, routing of the tow trains; third, setting a timetable for each tow train; fourth, determining the load for each tour. Battini et al. (2010) tackle the most long-term first problem of siting the decentralized logistics areas within the factory by considering part commonality, demand rates, inventory cost, and other factors. Similarly, Emde and Boysen (2012b) propose a continuous location model and, on the basis of aggregate demand data, they determine the optimal number of supermarkets, their location as well as the assignment of stations to supermarket via an exact polynomial-time dynamic programming scheme. The second problem (routing) is analyzed by Vaidyanathan et al. (1999), who, assuming constant demand rates and cyclic schedules, formulate a modified vehicle routing problem, which they call the Just-in-Time Capacitated Vehicle Routing Problem (JITCVRP). They present two heuristics to solve it. On the other end of the spectrum, (Emde et al., 2012) investigated the most short-term fourth problem by developing an exact polynomial-time algorithm to determine which bins to load for which station on each tour for a given schedule on a given route, minimizing both the total sum of the inventory as well as the maximum inventory at any one station.

Regarding the third planning step, the scheduling of the tow train, which is the central focus of this paper, it has so far mostly been tackled only as a simplified subproblem of other problems. Choi and Lee (2002) look at a comprehensive planning process that entails the routing, scheduling as well as loading problems. Their goal is to minimize the total deviation of actual from planned delivery times, where both earliness as well as lateness are undesirable, thus allowing for just-in-time concerns. They use mainly modified classic vehicle routing methods, specifically the insertion heuristic, to solve the problem, which are tested in a simulation experiment on real-world data from a South Korean assembly plant. Golz et al. (2012) similarly present a case study from a German automobile producer, solving the routing, scheduling and loading problems via an adapted version of the classic savings heuristic. Their aim is the minimization of the number of tow train operators to feasibly supply the stations. An exact polynomial-time approach combining both routing as well as scheduling is developed by Emde and Boysen (2012a), albeit under rather restrictive assumptions, such as that the tow train always stops at all stations on each tour and that parts are only refilled once all previously delivered ones are exhausted (zero inventory property). In a similar vein, Kilic and Durmusoglu (2013) seek to heuristically route and schedule tow trains, assuming cyclic

deliveries and constant demand rates. Fathi et al. (2014) tackle the scheduling problem more directly, allowing to delete tours from an otherwise already fixed timetable. Their goal is to minimize both the number of tours as well as the inventory at the line, using a procedure based on simulated annealing.

In many assembly plants, routing the tow train is not necessarily a very immediate concern because due to the very confined space and limited number of driving lanes on the shop floor, few possible routes are available anyways. Often a tow train is simply assigned to one of the available routes and the problem to be solved on a regular (e.g., once per shift or day) basis is the timetabling, i.e., when should the tugger leave the supermarket with what load, stopping at which stations? This specific problem has, to the very best of the authors' knowledge, never been addressed before. However, the problem bears a certain resemblance to single-machine scheduling, where the tow train can be seen as a machine and the tours it executes as jobs. Unlike typical machine scheduling formulations, jobs do not have fixed release and due dates, however, but rather depend on the demand at the stations which varies over time. Moreover, the processing times of the jobs (tours) are not fixed but depend on the number of stops. While machine scheduling problems with variable processing times have been studied in the literature in the form of so-called malleable tasks (e.g., Blazewicz et al., 2004), they assume that the processing time varies with the number of processors that handle them, which does not reflect the nature of our scheduling problem.

3 Problem description

The tow train scheduling and loading problem (TTSL) is concerned with determining the schedule as well as the load of a transport vehicle (tow train) carrying kits of parts, packaged in uniform standard-size bins, to a set S of workstations along a mixed-model assembly line. The tow train sets off from a central supermarket to visit these stations in a fixed order. Note that a given tow train route usually does not cover all stations of the entire assembly line but only a segment of the line, which has been determined in a previous planning step or is necessitated by the shop floor layout. Depending on the production sequence, detailing exactly which model is assembled at which station in which work cycle, in each cycle t in the planning horizon of length T either a new kit of parts is required at station s ($d_{st} = 1$) or not ($d_{st} = 0$). The sequence is usually fixed a few days before production starts, therefore the demands d_{st} can be assumed to be given and known with certainty (Emde and Boysen, 2012a). Note that since all bins are of homogeneous, standardized size (e.g., Golz et al., 2012, Emde et al., 2012), the exact contents of the bins is immaterial for the purposes of the TTSL; only the correct *number* of bins must always be delivered to the right stations. Putting the right parts into these containers is the concern of the pickers at the supermarket.

Travelling from the supermarket to station s takes a certain amount of time r_s, while r^{SM} stands for the driving time for a complete round trip, i.e., starting from the supermarket, driving by all stations on the route, and returning. Further, it takes some time P for the tow train to be replenished at the supermarket, and if the tugger makes

a stop at a station, a stopover time of p passes, during which the bins loaded on the vehicle are unloaded and empty bins are collected. Note that we assume here that the stopover time p is independent of the number of bins to be (un)loaded. This is certainly justified in (partially) automated delivery systems using shooter racks (e.g., Emde and Boysen, 2012a), where all bins are swapped in one go once the tow train has docked. Even in purely manual systems, however, using average stopover times is often a valid simplification, as the time to stop the vehicle, get off, start exchanging bins, and then get on again often dominates the relatively short time it takes to actually put a few more or less containers onto the rack, especially since deviations from the average are rarely more than a small handful of bins.

If the tugger does not stop at some station s on a given tour, then no stopover time at that station elapses; the vehicle still has to pass the station, however, incurring the travel time encoded in parameter r. All times are normalized to the cycle time, meaning, e.g., that a stopover time of $p = 0.5$ would correspond to half a minute if the cycle time is 60 seconds. All time related parameters can be fractional; however, for convenience's sake, we assume that the tow train only sets off at, and unloaded bins only become available at, integral points in time (i.e., full work cycles). Due to the rather short cycle times of about 60 to 90 seconds in the automotive industry (Emde et al., 2012), this seems like a tolerable simplification. Moreover, from a practical perspective, it would be very difficult to determine the exact point in time when demand occurs with any greater precision than this, or to communicate continuous timetables to the operators.

Due to the narrow driving lanes and sharp turns on typical shop floors, tow trains are usually restricted to just a handful of wagons and are thus limited to carrying a maximum number of C bins per tour. Space at the stations is also notoriously scarce (Bozer and McGinnis, 1992), therefore no more than c_s bins can be stored at any single time at station s.

We can now define a schedule Ω as follows. Each tour $k = 1, \ldots, n$ on the given route can be expressed as a $(|S| + 1)$-tuple $(\tau_k; z_1^k, z_2^k, \ldots, z_{|S|}^k) \in \Omega$, where $\tau_k \in \mathbb{N}^{/0}$ denotes the work cycle when the tugger sets off from the supermarket and $z_s^k \in \mathbb{N}^0$ is the load (i.e., the number of bins) that the tow train carries to station s on tour k. Note that $z_s^k > 0$ implies that the tow train makes a stop at station s on tour k, while $z_s^k = 0$ means that it will pass the station by. Moreover, note that the number of tours n is also variable. We say a schedule Ω is feasible if it fulfills the following conditions.

- For each tour $k = 2, \ldots, n$ it must hold that

$$\tau_k \geq \left\lceil \tau_{k-1} + r^{SM} + P + \sum_{\substack{s \in S: \\ z_s^{k-1} > 0}} p \right\rceil,$$

  i.e., tour k can only start after the preceding tour $k - 1$ has finished.

- The last tour n must end within the planning horizon:

$$\tau_n + r^{SM} + \sum_{\substack{s \in S: \\ z_s^n > 0}} p \leq T.$$

- For each tour $k = 1, \ldots, n$, it must hold that $\sum_{s \in S} z_s^k \leq C$, i.e., the total load in tour k cannot be greater than the vehicle capacity C.

- Let

$$\delta_{sk} = \left\lceil \tau_k + r_s + \sum_{\substack{s' \in S: \\ s' \leq s \wedge z_{s'}^k > 0}} p \right\rceil$$

be the work cycle when the bins (if any) unloaded at station s on tour k become available. Furthermore, let

$$l_{st} = \sum_{\substack{k=1: \\ \delta_{sk} \leq t}}^{n} z_s^k - \sum_{t'=1}^{t} d_{st'} + l_s^0$$

be the number of bins in stock at station s in cycle t, where l_s^0 is the initial inventory at station s. Then

$$0 \leq l_{st} \leq c_s$$

must hold, $\forall s \in S$, $t = 1, \ldots, T$, i.e., no stock-outs must occur and no station must be overstocked.

t	1	2	3	4	5
d_{1t}	0	0	0	1	1
d_{2t}	0	0	1	0	1
d_{3t}	0	0	1	1	0
l_{1t}	0	1	1	0	0
l_{2t}	0	2	1	1	0
l_{3t}	0	0	1	0	0

Table 1: Example demands and inventory.

Apart from finding a schedule that feasibly feeds parts to the workstations without any station ever starving for parts, one of the most pressing concerns of plant managers is typically in-process inventory. Practically all automobile producers aim at an inventory strategy in line with the famous just-in-time principle, trying to reduce, if not wholly eliminate, line-side stock. Accordingly, in the TTSL we seek among all feasible schedules

the one where the total amount of bins lying in stock $f(\Omega) = \sum_{s \in S} \sum_{t=1}^{T} l_{st}$ is minimal.

Example: Consider the example demands from Table 1 for $|S| = 3$ stations and $T = 5$ work cycles. Let the replenishment time in the supermarket be $P = 1$, the stopover time per station $p = 0.3$, and the driving times from the supermarket to the stations $r_1 = 0.1$, $r_2 = 0.2$, $r_3 = 0.4$, and finally $r^{SM} = 0.5$ for a complete round trip. Let the maximum capacity of the tow train be $C = 5$ and that of the stations $c_1 = 3$, $c_2 = 3$, and $c_3 = 2$, respectively, and the initial inventory $l_s^0 = 0$, $\forall s \in \{1, 2, 3\}$. Then the optimal schedule Ω is $\{(1; 1, 2, 2), (4; 1, 0, 0)\}$, leading to $\delta_{1,1} = 2$, $\delta_{2,1} = 2$, $\delta_{3,1} = 3$, $\delta_{1,2} = 5$, and an objective value of $f(\Omega) = 7$. Detailed information about the bins in stock l_{st} at each station in each work cycle can be found at the bottom of Table 1.

4 Time complexity

In this section we will investigate the computational complexity of TTSL, as well as of a few special cases. First, we will analyze the general problem, as described in the previous section.

Proposition 4.1. *Finding a feasible solution to TTSL is strongly NP-complete.*

Proof. Membership in NP is easy to see: Tour start times τ_k and loads per tour per station z_s^k constitute a certificate, which can clearly be verified in polynomial time if we restrict ourselves to the case where $n \leq T$. Note that more than one tour per cycle is impossible if any of the time-related parameters, i.e., driving time, stopover time, and/or replenishment time, is > 0.

We will show that the problem is NP-hard by pseudo-polynomial transformation from 3-PARTITION, which is well known to be NP-hard in the strong sense (Garey and Johnson, 1979).

An instance of 3-PARTITION is defined as follows. Given $3q$ positive integers a_j $(j = 1, \ldots, 3q)$ and a positive integer B with $B/4 < a_j < B/2$ and $\sum_{j=1}^{3q} a_j = qB$, does there exist a partition of the set $\{1, 2, \ldots, 3q\}$ into q sets $\{A_1, A_2, \ldots, A_q\}$, each having exactly three elements, such that $\sum_{j \in A_i} a_j = B$, $\forall i = 1, \ldots, q$?

We consider a special case of 3-PARTITION where B is divisible by 3. This problem variant is obviously also NP-hard since any 3-PARTITION instance can be transformed by simply multiplying by 3 B and each a_j. Such an instance can be transformed to an instance of TTSL in pseudo-polynomial time in the following way. The number of work cycles in the planning horizon is $T = B + 2q + 3$, the stopover time is $p = 1/3$ work cycles, the replenishment time is $P = 1$, and the driving times and initial inventories are all 0. For each of the $3q$ integers in the original problem introduce one station s_j of type 1. The demand for bins at these stations is $d_{s_j,t} = 1$, $\forall t = 2q, \ldots, 2q + a_j - 1$, $j = 1, \ldots, 3q$, and otherwise 0, such that the total cumulated demand at each station of type 1 sums

up to exactly the corresponding integer from 3-PARTITION, a_j. Furthermore, $3B$ stations of type 2 are introduced, which only demand one single bin at time $B + 2q + 3$: $d_{s_j,B+2q+3} = 1, \forall j = 3q + 1, \ldots, 3q + 3B$. All stations of type 1 and 2 combined form the set of stations S of the transformed TTSL instance. Finally, the tow train capacity is $C = B$, and the station capacities can assume any value $c_s \geq B/2$.

A feasible solution to such a TTSL instance can be transformed in pseudo-polynomial time to a solution to the corresponding 3-PARTITION instance and vice versa. The total number of stations to be visited equals $|S| = 3q + 3B$. There is no driving time to consider, but since the stopover time at each station is $p = 1/3$, it will take the tow train at least $q + B$ time units to stop at all stations. Furthermore, even if the tow train is always loaded to the brim, it will still have to make at least $(qB + 3B)/C = q + 3$ stops at the supermarket to replenish, taking $P = 1$ time unit each time. Seeing that the last tour must finish within the planing horizon of $T = B + 2q + 3$, the tugger must obviously be constantly busy without any idle time whatsoever and must visit every station exactly once.

Given this, visiting all the stations of type 1 takes exactly $2q$ work cycles (q cycles for stopovers plus q cycles for replenishment), and visiting all stations of type 2 takes exactly $B + 3$ cycles (B cycles for stopovers plus 3 cycles for replenishment). Now, the tow train must first finish all deliveries to the stations of type 1 before it can start making deliveries to the other stations because the first demand at the type 1 stations occurs at time $2q$ and supplying these stations takes $2q$ time units. Seeing that the tow train can only make one stop per station and it cannot load more than $C = B$ bins per tour, it must obviously visit the stations of type 1 in groups of three, supplying all their demand at once, which is only possible if the train is always fully loaded with B bins. The correspondence to the 3-PARTITION solution is thus easy to see. □

The above proof also allows making the following observation.

Corollary 4.1. *Finding a feasible solution to TTSL is NP-complete in the strong sense even if the driving times are 0, the station capacities are infinite, and the departure times are fixed.*

Proof. That the driving times and capacity restrictions at the stations are immaterial for the logic of the preceding proof is obvious. The schedule, i.e., the departure times of the tow train τ_k, can also be fixed like imposed by the logic of the proof (one departure every 2 time units up until time $2q - 1$, and then three departures at time $2q + 1$, $2q + 2 + 1/3 \cdot B$, and $2q + 3 + 2/3 \cdot B$) without making the problem any easier. □

Now, we will take a closer look at one important special case that does indeed make the problem more tractable, namely if there are no station capacities and stopover times to consider. While the stopover time will obviously never be zero in reality, this may nonetheless be a reasonable simplification for fully automated systems, where the stopover time may be negligible compared to the driving time (Emde et al., 2012). As to the station capacities, seeing that the objective function minimizes the total inventory at

the line, it stands to reason that the maximum inventory at any one station will probably not grow excessively large, either. If shelf space is not too big an issue, assuming infinite station capacities may be a valid relaxation. Finally, even if these conditions are not satisfied for a given production setting, a solution of the relaxed problem should at least provide a fairly good initial solution for further optimization.

Proposition 4.2. *TTSL can be solved to optimality in polynomial time if the stopover time is zero, i.e., $p = 0$, and the station capacities c_s are infinite.*

Proof. In order to prove the proposition, we will construct a polynomial-time exact algorithm based on dynamic programming (DP). First of all, note that, due to the infinite capacity at the stations, the initial inventory I_s^0 plays no role for this special case, because it can simply be subtracted from the demand d_{st}. The consequent unavoidable inventory can then simply be added to the optimal objective value once the DP has run its course because it is constant and does not depend on the tow train schedule at all.

The DP is subdivided into $\overline{n} + 2$ stages, each stage representing one tour, where $\overline{n} = \lfloor T/\lceil P + r^{SM} \rceil \rfloor$ is the theoretical maximum number of tours. The stages are successively constructed starting from a (virtual) start stage 0. Each stage consists of states (τ, w), where $\tau \in \{0, \ldots, T+1\}$ is the departure time of the tow train from the supermarket and $w \in \{0, 1, \ldots, \sum_{s \in S} \sum_{t=1}^{T} d_{st}\}$ is the "surplus" of bins the tow train has transported to the stations that have not yet been consumed. In other words, w denotes the number of bins that are "carried over" (by being stocked at the stations) from previous tours into the current tour. Note that states $(0, 0)$ and $(T+1, 0)$ are the (virtual) first and last states in the DP graph, respectively.

These pieces of information are enough to fully describe a schedule if $p = 0$. Vehicle capacity permitting, in tour k the tow train will bring exactly the number of bins required up until its next arrival in tour $k+1$ (or the end of the planning horizon if k is the last tour). This is obviously optimal for a given schedule since deliveries cannot possibly be any more just-in-time than this. There is no need to keep track of the individual stations at which the demand occurs; given that the stopover time is 0, the tugger can stop at any station as necessary without upsetting the schedule in any way. However, there is, of course, no guarantee that the vehicle capacity is not a bottleneck. In this case, the bins that exceed the capacity and cannot be brought just-in-time must have been stocked in a previous tour, hence the need to keep track of the "surplus" bins w. However, again, it is pointless to specifically allocate these bins to individual stations, it is sufficient to know that the *total* number of bins delivered up to a point in time is enough to cover the total demand up to that point, since the tow train can distribute the bins as required because it can visit any station in any tour without further consequences.

From dummy start state $(0, 0)$ of stage 0 successor states $(\tau, 0)$ of stage 1 are reached for all $\tau \in \left\{ t \mid \sum_{s \in S} \sum_{t'=0}^{t} d_{s,t+\lceil r_s \rceil} = 0; t = 1, \ldots, T \right\}$. Since the start state only represents a "dummy tour" with the sole purpose of determining the start time of the real first tour, it cannot supply any demand and hence no demand must occur before the start time of the actual first tour represented by the states of stage 1. Then, the following transitions lead from a state (τ, w) of stage k to another state (τ', w') of stage $k+1$, for all $1 \leq k \leq \overline{n}$.

- Add another tour to the schedule: $(\tau, w) \rightarrow (\tau', w')$, where $\tau' \in \{\lceil \tau + \tau^{SM} + P \rceil, \ldots, \lfloor T - \tau^{SM} \rfloor\}$ and $w' \in \left\{\max\left\{0, w - \sum_{s \in S} \sum_{t=\tau+\lceil r_s \rceil}^{\tau'+\lceil r_s \rceil - 1} d_{st}\right\}, \ldots, w + C - \sum_{s \in S} \sum_{t=\tau+\lceil r_s \rceil}^{\tau'+\lceil r_s \rceil - 1} d_{st}\right\}$.
- Make the current tour the last one: $(\tau, w) \rightarrow (T+1, 0)$, which is only possible if $C + w \geq \sum_{s \in S} \sum_{t=\tau+\lceil r_s \rceil}^{T} d_{st} \geq w$, i.e., the vehicle capacity plus "surplus bins" are sufficient to meet all the remaining demand, and all surplus bins are used up.

Let $\Gamma(\tau', w')$ be the set of states from which a transition to state (τ', w') exists. Then the (partial) objective value $G(\tau', w')$ can be calculated as follows.

$$G(\tau', w') = \min_{(\tau, w) \in \Gamma(\tau', w')} \left\{ G(\tau, w) + (\tau' - \tau) \cdot w + \sum_{s \in S} \sum_{t=\tau+\lceil r_s \rceil}^{\tau'+\lceil r_s \rceil - 1} \left(\sum_{t'=\tau+\lceil r_s \rceil}^{\tau'+\lceil r_s \rceil - 1} d_{st'} - \sum_{t'=\tau+\lceil r_s \rceil}^{t} d_{st'} \right) \right\},$$

where $G(0,0) := 0$. The first term represents the partial objective value up to predecessor state (τ, w), the second term is the contribution of the "surplus" bins w that are carried over into the current tour, and the third term is the contribution of the bins that are consumed "just-in-time," before the next arrival of the tow train, and are hence not carried over.

Input: List of states $\langle (\tau_0, w_0), \ldots, (\tau_{n+1}, w_{n+1}) \rangle$ constituting the optimal path in the DP graph

$u_s := 0, \forall s \in S$;
for $k = n+1$ **down to** 2 **do**
  $c := C$;
  foreach $s \in S$ **do**
    $D := \sum_{t=\tau_{k-1}+\lceil r_s \rceil}^{\tau_k+\lceil r_s \rceil - 1} d_{st}$;
    $z_s^{k-1} := \min\{D, c\}$;
    $c := c - z_s^{k-1}$;
    $D := D - z_s^{k-1}$;
    $u_s := u_s + D$;
  end
  foreach $s \in S$ **do**
    $z_s^k := z_s^{k-1} + \min\{u_s, c\}$;
    $c := c - z_s^{k-1}$;
    $D := D - z_s^{k-1}$;
  end
end

Output: optimal tow train loads z_s^k

Algorithm 1: Backward recovery of the optimal solution.

The optimal solution is represented by end state $(T, 0)$ with optimal objective value $G(T, 0)$. The optimal loads of the tow train can be obtained via backward recovery along the optimal path (Algorithm 1), the basic idea being that in a tour k, the tow train will

first try to service all demand until the next tour just-in-time (the first for-loop). If that fails due to insufficient capacity, the bins exceeding the capacity have to be delivered by an earlier tour (stored in variables u_s). Conversely, if the tow train still has capacity left after satisfying all just-in-time demand, the surplus capacity is used to service the leftover demand from later tours (second for-loop).

Concerning the asymptotic runtime, the DP consists of $\overline{n} + 2$ stages, where we can safely assume that the maximum number of tours n is bounded by the number of work cycles T. In each stage there are at the very most $T \cdot \sum_{s \in S} \sum_{t=1}^{T} d_{st}$ states. Since no more than one bin can be in demand per station per cycle, the total number of states is thus bounded by $T^3 \cdot |S|$. Even if every state were connected to every other state, the total number of transitions would still be in $O(T^6 \cdot |S|^2)$, while the objective value per transition can be computed in quadratic time. Therefore, the proposition holds. □

Example (cont.): Consider the example from Section 3, except with $p = 0$. Figure 2 shows the corresponding DP graph, leading to the optimal solution (bold in the figure) with objective value 2, where the tow train starts a tour at time 2 and again at time 4, carrying 1 bin each to stations 1 and 2 and 2 bins to station 3 in the first tour, and then one bin each to stations 1 and 2 in the second tour.

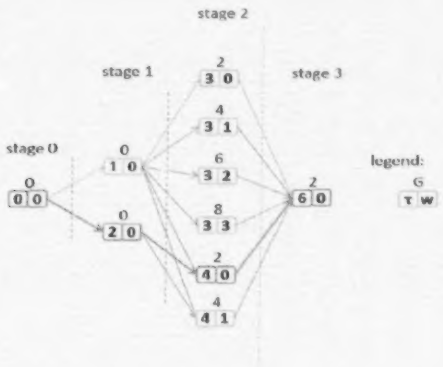


Figure 2: DP graph for the example.

5 Algorithms

In this section, we will investigate how to efficiently solve TTSL. Since even the feasibility version of this problem is NP-hard in the strong sense (Proposition 4.1), heuristics will most probably be necessary to solve instances of realistic size in acceptable time. Nonetheless, in order to have a benchmark, we also propose a mixed-integer programming formulation that can be used by a default solver to obtain guaranteed optimal

results. Then, we will present decomposition scheme that we will use as part of a tabu search heuristic.

5.1 MIP model

Using the additional notation from Table 2, the TTSL can be formulated as a MIP model as follows.

$$\text{Minimize } \mathcal{F} = \sum_{t-1}^{T} \sum_{s \in S} l_{st} \tag{1}$$

subject to

$$\sum_{t-1}^{T} \sum_{s \in S} z_{kst} \leq C \qquad \forall k = 1, \ldots, n \tag{2}$$

$$0 \leq l_{st} \leq c_s \qquad \forall t = 1, \ldots, T, s \in S \tag{3}$$

$$l_{st} + d_{st} - \sum_{k=1}^{n} z_{kst} = l_{s,t-1} \qquad \forall t = 1, \ldots, T, s \in S \tag{4}$$

$$l_{s0} = l_s^0 \qquad \forall s \in S \tag{5}$$

$$y_{kst} \leq z_{kst} \leq y_{kst} \cdot C \qquad \forall k = 1, \ldots, n, s \in S, t = 1, \ldots, T \tag{6}$$

$$y_{kst} \leq f_k \qquad \forall k = 1, \ldots, n, s \in S, t = 1, \ldots, T \tag{7}$$

$$\tau_k - \tau_{k-1} - \sum_{s \in S} \sum_{t=1}^{T} y_{k-1,s,t} \cdot p - (r^{SM} + P) \cdot f_k \geq 0 \qquad \forall k - 2, \ldots, n \tag{8}$$

$$\tau_k + \sum_{s \in S} \sum_{t=1}^{T} y_{kst} \cdot p + r^{SM} \cdot f_k \leq T \qquad \forall k = 1, \ldots, n \tag{9}$$

$$t \cdot y_{kst} - \tau_k - r_s - 1 + \epsilon - \sum_{\substack{s' \in S; \\ s' \leq s}} \sum_{t'=1}^{T} y_{ks't'} \cdot p \leq 0 \qquad \forall k = 1, \ldots, n; s \in S; t = 1, \ldots, T \tag{10}$$

$$t + (1 - y_{kst}) \cdot M - \tau_k - r_s - \sum_{\substack{s' \in S; \\ s' \leq s}} \sum_{t'=1}^{T} y_{ks't'} \cdot p \geq 0 \qquad \forall k = 1, \ldots, n; s \in S; t = 1, \ldots, T \tag{11}$$

$$y_{kst} \in \{0; 1\} \qquad \forall k = 1, \ldots, n, s \in S, t = 1, \ldots, T \tag{12}$$

$$f_k \in \{0; 1\}, \tau_k \in \mathbb{N}^{\neq 0} \qquad \forall k = 1, \ldots, n \tag{13}$$

Objective function (1) seeks to minimize the total inventory. Constraints (2) make sure that the tow train is not overloaded and (3) make it impossible that a station starves for parts or exceeds its maximum shelf space. Inventory that is not consumed flows to the next period (4), and the initial inventory equals l_s^0 (5). Constraints (6) enforce that inventory can only be replenished at station s at time t if the tow train actually stops at the station at that time, while also prohibiting "empty" stops, when no bins are delivered. The tow train can only make stops on a given tour if the tour actually takes place, which is ensured by (7). (8) render overlapping tours impossible and (9) ensure that the last tour ends within the planning horizon of T work cycles. Note that (8) and (9) are only meaningful for tours that actually take place ($f_k = 1$). Inequalities (10) and (11) set the work cycle when the bins delivered on tour k to station s become available (encoded in variables y_{kst}) no sooner and no later than the time when the tow train actually reaches them, rounded up to the next full work cycle. Finally, (12) and (13) define the domain of the variables.

M	big integer
ϵ	very small real number
l_{st}	continuous variable: amount of unconsumed bins in stock at station s at time t
z_{kst}	continuous variable: number of bins the tow train takes to station s at time t on tour k
y_{kst}	binary variable: 1, if the bins delivered to station s on tour k become available at time t; 0, otherwise
τ_k	integer variable: work cycle when the tow train sets off from the supermarket on tour k
f_k	binary variable: 1, if tour k takes place; 0, otherwise

Table 2: Notation

5.2 Heuristic decomposition of TTSL

Due to the NP-hard nature of the problem, default solvers and other exact solution tools will most likely be too slow to be useful for solving instances of real-world size. We will therefore propose a heuristic decomposition scheme that will serve as the basis of a neighborhood search method.

The MIP-model proposed in the previous subsection is difficult to solve mainly because of the large number of binary variables y_{kst}, indicating whether or not bins become available at station s in work cycle t delivered on tour k. This essential piece of information can be broken down into two basic components: First, when does the tow train leave the supermarket, and second, at which stations does it make a stop on each tour? Once these two questions are answered, the detailed delivery schedule $\overline{y}_{kst}$ can easily be computed, and so can the exact optimal load of the tow train, as we will show in a moment.

Note that the two questions cannot be answered completely independently of each other because the number of stopovers per tour influences the total trip time of the vehicle and consequently the feasible departure times. Instead of using the departure time τ_k as a decision variable, it seems therefore more meaningful to encode the time-related aspect as an offset $b_k \in \mathbb{N}^0$, indicating how many work cycles pass in-between the end of tour $k-1$ and tour k (or in-between time 1 and the very first departure of the tugger in case of $k = 1$). Further, an array of binary variables o_{sk} encoding whether or not the tow train stops at station s on tour k is required. Consequently, a solution Σ for our neighborhood search is encoded as an array of length $n + n \cdot |S|$, containing both the buffers in-between tours b_k as well as the binary stopover variables o_{sk}. The original variables $\overline{y}_{kst}$ can then easily be reconstructed: Let

$$\delta_{sk} = \sum_{k'=1}^{k-1} \left\lceil P + r^{SM} + \sum_{s' \in S} o_{s'k'} \cdot p + b_{k'} \right\rceil + \left\lceil b_k + r_s + \sum_{\substack{s' \in S: \\ s' \leq s}} o_{s'k} \cdot p \right\rceil$$

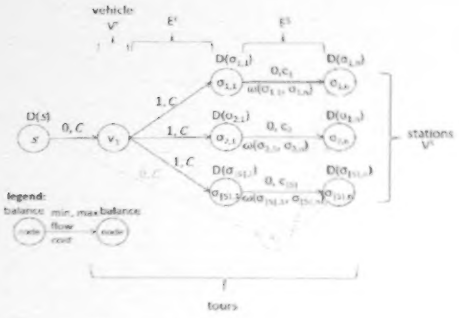
be the work cycle when the bins delivered to station s on tour k become available. Then

$$\overline{y}_{kst} = \begin{cases} 1 & \text{if } o_{sk} = 1 \wedge t = \delta_{sk} \\ 0 & \text{else.} \end{cases}$$

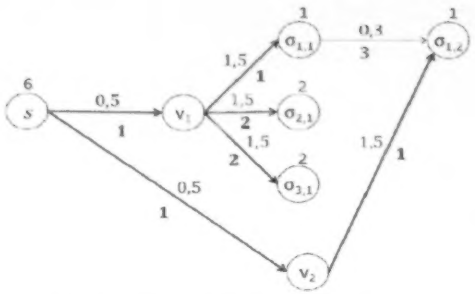
Once the binary variables are fixed, the loading problem can be formulated as a normal linear programming model (LP), with objective function (1), subject to (2) - (6), where variables y_{kst} are replaced with fixed values $\overline{y}_{kst}$. All remaining variables z_{kst} and l_{st} are continuous; they will always assume integral values in the optimal solution. The loading subproblem can thus be solved in polynomial time.

In fact, this subproblem can even be solved in strongly polynomial time using the network simplex algorithm (Orlin, 1997) if it is formulated as a minimum cost network flow problem. Consider a digraph $\mathcal{G}(s \cup V^t \cup V^S, E^t \cup E^S)$, where s is the source, while node set $V^t = \{v_1, \ldots, v_n\}$ stands for the tours by the vehicle, and $V^S = \{\sigma_{sk} \mid s \in S \wedge k = 1, \ldots, n \wedge o_{sk} = 1\}$ for the stations during each tour. Source s has a supply equalling the total demand, i.e., $D(s) = \sum_{s \in S} \left(\sum_{t=1}^{T} d_{st} - l_s^0 \right)$. Each sink node $\sigma_{sk} \in V^S$ is associated with a demand equal to the bin demand between tours, i.e., $D(\sigma_{sk}) = \min \left\{ 0; -\sum_{t=\delta_{sk}}^{\delta_{sk'}-1} d_{st} + \max \left\{ 0; l_s^0 - \sum_{t=1}^{\delta_{sk}-1} d_{st} \right\} \right\}$, where k and k' are consecutive tours that stop at station s (if k is the last tour to stop at station s, then $\delta_{sk'} := T + 1$).

Each tour node $v_k \in V^t$ is connected to source s via an arc $(s, v_k) \in E^t$, with maximum capacity $\overline{c}(s, v_k) = C$, thus ensuring that the vehicle is never overloaded. Furthermore, there is an arc $(v_k, \sigma_{sk}) \in E^t$ for each $\sigma_{sk} \in V^S$, signifying the bins transferred on tour k to station s. Note that, due to the definition of V^S, a given node σ_{sk} only exists if station s actually lies on tour k. In that case, the minimum flow on this arc is $\underline{c}(v_k, \sigma_{sk}) = 1$, enforcing the impossibility of "empty" stops. An upper bound on the flow is not really necessary for these arcs but can be set to $\overline{c}(v_k, \sigma_{sk}) = C$. Moreover, there is an arc $(\sigma_{sk}, \sigma_{sk'}) \in E^S$ iff $k' > k$ and $o_{sk} = 1$ and $o_{sk'} = 1$ and $\forall k'' = k+1, \ldots, k'-1 : o_{sk''} = 0$,



(a) Concept of the flow network.



(b) Specific flow network from the example; edges with non-zero flow in the optimal solution are bold.

Figure 3: Flow networks.

meaning that the tugger makes two consecutive stops at station s on tour k and k' (but not in the meantime), denoting the inventory that stays at the station until the next arrival of the tow train. Each arc $(\sigma_{sk}, \sigma_{sk'}) \in E^S$ has a maximum capacity restricted by the station capacity, namely $\overline{c}(\sigma_{sk}, \sigma_{sk'}) = c_s$.

Each arc $(\sigma_{sk}, \sigma_{sk'}) \in E^S$ is associated with a cost $\omega(\sigma_{sk}, \sigma_{sk'}) = \delta_{sk'} - \delta_{sk}$ per unit of flow. This means that each unit of flow sent over arc $(\sigma_{sk}, \sigma_{sk'})$ will increase the cost proportionally to the number of cycles that each additional bin will have to be stored at the station. Note that if there is no flow over some arc $e \in E^S$, then this signifies that all bins in demand between tour k and k' are supplied "just-in-time", with no additional bins stored at the station. A positive flow F, on the other hand, implies that $F(e)$ bins are stocked at station s for consumption later than the tow trains subsequent arrival $\delta_{sk'}$. The total cost to be minimized is then

$$f^{net} = \sum_{e \in E^S} \omega(e) \cdot F(e) + \sum_{s \in S} \sum_{\substack{k=1: \\ o_{sk}=1}}^{n} \sum_{t=\delta_{sk}}^{\delta_{sk'}-1} \left(\sum_{t'=\delta_{sk}}^{\delta_{sk'}-1} d_{st'} - \sum_{t'=\delta_{sk}}^{t} d_{st'} \right), \tag{14}$$

where the first term denotes the storage cost as explained above, whereas the second term is the unavoidable inventory determined by the given schedule, where $\delta_{sk'}$ denotes either the time of the earliest stop at station s later than δ_{sk}, or $T + 1$ if the last stop occurred during tour k. Note that the flow (and thus the solution to the network flow problem) has no bearing on this second term. Therefore, solving the minimum-cost network flow problem essentially only determines the amount of inventory over and above the unavoidable "just-in-time" inventory, that is, the inventory that is carried over from one tour to the next. The whole network is schematically depicted in Fig. 3a.

Example (cont.): Assume that for our example problem from Section 3 $n = 2$ tours have been fixed, with time buffers $b_1 = b_2 = 0$ (meaning both tours start as early as

possible) and stopovers $o_{1,1} = o_{2,1} = o_{3,1} = o_{1,2} = 1$ (meaning that the tow train is to stop at all stations on its first tour and only at station 1 on its second tour). The ensuing flow network – with optimal flows boldfaced – is in Figure 3b. This optimal network flow solution corresponds to the original solution from Table 1 with objective value $f^{net} = f(\Omega) = 7$.

The optimization can hence be split into two parts: One, finding a good (or even optimal) solution vector of offsets b_k and stopovers o_{sk} (the master problem), and, two, determining the optimal loads for the given schedule (the slave problem). There is, however, one potential issue with solving the problem in this manner: There is no guarantee that a feasible load even exists for a given schedule. In order to get some feedback from the subproblem about not just whether or not a schedule is infeasible but also about *how* infeasible it is, some information on the feasibility of the schedule is obtained prior to solving the network flow problem, in hopes that the overarching optimization process can then use this information to steer the search towards the feasible regions of the solution space.

- The minimum amount of bins that must be stored at station s for a given schedule is $\underline{d}_s = \max_{k=1,\dots,n} \left\{ \sum_{t=\delta_{sk}}^{\delta_{sk'}-1} d_{st} - \max\left\{0; \ell_s^0 - \sum_{t=1}^{\delta_{sk}-1} d_{st}\right\}\right\}$, where $\delta_{sk'}$ is the time of the next stop at station s after tour k or $T+1$ if there are no more stops. If $\underline{d}_s > c_s$, the schedule is infeasible due to overloaded racks, and the "degree" of infeasibility is $g_s^c = \max\{0, \underline{d}_s - c_s\}$, $\forall s \in S$.
- For each tour $k = 1, \dots, n$, the cumulated number of bins that must have been delivered up to and including that tour is $\underline{C}_k = \sum_{s\in S}\left(\sum_{t=1}^{\delta_{sk'}-1} d_{st} - \ell_s^0\right)$. If $\exists k = 1, \dots, n : \underline{C}_k > C \cdot k$ then the schedule is infeasible due to insufficient vehicle capacity, incurring $g_k^C = \max\{0, \underline{C}_k - C \cdot k\}$, $\forall k = 1, \dots, n$.
- If $e^n = \sum_{k=1}^{n} \lceil P + r^{SM} + \sum_{s\in S} o_{sk} \cdot p + b_k \rceil + \lceil b_n + r^{SM} + \sum_{s\in S} o_{sn} \cdot p \rceil > T$, the schedule is infeasible because it does not end within the planning horizon, hence $g^n = \max\{0, e^n - T\}$.
- For each station $s \in S$, let $\underline{\delta}_s = \min_{k=1,\dots,n}\{\delta_{sk} | o_{sk} = 1\}$ be the point in time when the tow train makes its first delivery to station s if it makes any stop at all, i.e., if $\sum_{k=1}^{n} o_{sk} > 0$; otherwise let $\underline{\delta}_s = T+1$. If $\underline{t}_s = \min_{t=1,\dots,T}\{t | \sum_{t'=1}^{t} d_{st'} > \ell_s^0\} < \underline{\delta}_s$, then the schedule is infeasible because the first demand that is not covered by the initial inventory occurs before the first delivery; $g_s^d = \max\{0, \underline{t}_s - \underline{\delta}_s\}$, $\forall s \in S$.

Note that even if no violations g have been detected, there is still no guarantee that a feasible load for the schedule can be found. Namely, either the vehicle or station capacity constraints could still be violated. We therefore add to our network $\mathcal{G}$ a "dummy" tour node v^d to node set V^t as well as an arc (s, v^d) with associated cost per unit of flow $\omega(s, v^d) = D(s) \cdot T$, that is, prohibitively high. Finally, v^d is connected to each node in $\sigma_{sk} \in V^S$ via arc (v^d, σ_{sk}). This way, if stock and regular deliveries are insufficient to meet demand, "dummy" bins are routed through node v^d, which is, however, prohibitively

expensive and will therefore only be done if there is no regular (i.e., feasible) way to supply the stations otherwise. Let g^p_{sk} be the total flow on arc (v^d, σ_{sk}). Then the total penalized objective value for a given schedule Σ is

$$f^{pen} := f^{net} + \sum_{s \in S} \left(\sum_{k=1}^{K} \eta^p_{sk} \cdot g^p_{sk} + \eta^c_s \cdot g^c_s + \eta^d_s \cdot g^d_s \right) + \sum_{k=1}^{K} \eta^C_k \cdot g^C_k + \eta^n \cdot g^n, \qquad (15)$$

where η^* is the penalty factor associated with violation g^*.

To speed up the search process, we only solve the network flow problem for those schedules Σ where $g^C_k = g^n = g^d_s = g^c_s = 0$, $\forall s \in S; k = 1, \ldots, n$. Otherwise, the network flow term $\sum_{e \in E^S} \omega(e) \cdot F(e)$ in Eq. (14) as well as g^p_{sk} is set to 0, $\forall s \in S; k = 1, \ldots, n$.

5.3 Neighborhood search

The components from the previous subsection can be used to design a metaheuristic neighborhood search scheme. The metaheuristic controls the overarching optimization by varying the schedule Σ, while the penalized objective value for each proposed schedule can be calculated as per Equation (15). An initial solution is obtained by solving the relaxed problem via the DP from Proposition 4.2.

Given an incumbent solution Σ, a neighbor Σ' is reached by way of one of the following moves.

stop move An additional stop is added to or a heretofore existing stop is removed from a tour, i.e., for some $k \in \{1, \ldots, n\}$ and $s \in S$, $o'_{sk} := 1 - o_{sk}$. A stop may also be inserted in a new tour, provided that the total number of tours does not exceed the theoretical maximum $\overline{n} = \lfloor T / \lceil P + r^{SM} + p \rceil \rfloor$. Similarly, if the last stop is removed from a tour, the whole tour is removed from the schedule, so long as the total number of tours is never below $\underline{n} = \left\lceil \sum_{t=1}^{T} \sum_{s \in S} d_{st} / C \right\rceil$.

time move The time buffer between two tours is increased or decreased by one work cycle, i.e., for some tour $k \in \{1, \ldots, n\}$, either $b'_k := b_k + 1$ or $b'_k := b_k - 1$, where a decrease is only possible if $b_k > 0$.

Finally, in order to evaluate objective function (15), the penalty factors η have to be set, which is done in a self-adjusting way (Hertz, 1992, Gendreau et al., 1994): If the last 5 accepted neighbors all were infeasible with regard to some constraint(s), i.e., $g^* > 0$ ($* \in \{p, c, C, n, d\}$), then the corresponding penalty factor(s) are changed to $\eta^* := 2 \cdot \eta^*$. If the last 5 accepted neighbors were all feasible with regard to constraint $*$, then $\eta^* := \eta^*/2$. Initially, $\eta^* := T \cdot |S|$, $\forall * \in \{p, c, C, n, d\}$. Note that accepting infeasible solutions is indispensable for solving TTSL, since even finding a feasible solution is a strongly NP-hard problem; disallowing infeasible solutions would make breaking out of local optima (or even finding any optima at all) nearly impossible.

As a neighborhood search-based metaheuristic approach we propose tabu search (TS) (Glover and Laguna, 1997), which has often been applied to complex scheduling problems

with difficult feasibility constraints to great success (e.g., Gendreau et al., 1994, Gendreau and Potvin, 2010). TS typically investigates the entire neighborhood of the incumbent solution in each iteration. For many problems, this leads to good solutions being found (e.g., Arostegui Jr et al., 2006, Gendreau et al., 1994).

For the TTSL, TS proceeds as follows. Starting from an initial solution Σ, all neighborhood solutions are generated, and the one with the lowest penalized objective value f^{pen} that is not tabu becomes the new incumbent solution to replace Σ. A solution, once visited, is made tabu for 500 iterations by adding its hash code

$$h(\Sigma) = \sum_{k=1}^{n} \left(b_k \cdot 2^{(|S|+\mathcal{B})\cdot(k-1)} + \sum_{s \in S} o_{sk} \cdot 2^{s-1+\mathcal{B}+(|S|+\mathcal{B})\cdot(k-1)} \right)$$

to a first-in-first-out queue of maximum length 500. Note that hash function h makes sure that each solution Σ gets a unique hash code, provided $\mathcal{B}$, the number of bits reserved for each b_k, is great enough, e.g., $\mathcal{B} := \lceil \log_2(T) \rceil$.

For the purpose of diversification, if no new globally best solution could be found for more than 100 iterations, Σ is replaced by a new (random) initial solution, generated as follows: First, an arbitrary number of tours n from the interval $\{\underline{n}, \ldots, \overline{n}\}$ (discrete uniform distribution) is selected. Then, o_{sk} is set to 1 with a likelihood of 20% if $\sum_{k'=1}^{k-1} o_{sk'} = 0$, i.e., if no earlier stop has been fixed yet; if an earlier stop already exists, the likelihood is 10%. The buffer between tours is $b_k := 0$, $\forall k = 1, \ldots, n$. Finally, the optimization ends after Θ seconds have elapsed, at which point the best found solution is returned.

To make better use of multi-core / multi-CPU workstations, we parallelized this algorithm: All neighbors are evaluated concurrently during each iteration, i.e., multiple network flow problems are solved in parallel.

6 Computational study

In this section, we present a numerical study in order to evaluate the computational performance of our proposed solution methods. To this end, we implemented our algorithms in C# 5.0 and ran them on an x64 PC equipped with an Intel Core i7-5500U 2.4 GHz CPU and 8,192 MB of RAM. Apart from performance metrics, we also compare our optimized schedules with simple cyclic schedules such as they are mostly used in current industrial practice, and investigate the benefit of employing automated guided vehicles. Because there are no established test data for TTSL, we will first elaborate how we generated our instances.

6.1 Instance generation

In accordance with our observations in practice, we generate the instance parameters the following way. First, we distinguish between small instances, where the number of stations served by the tow train is $|S| = 10$ and the planning horizon consists of $T = 24$ cycles, and large instances, where $|S| = 20$ and $T = 144$. Regardless of instance size, the

driving time from the depot to the first station is set to $r_1 = rnd(0.05, 0.2)$, where rnd is a uniformly distributed random number from the interval in the argument, rounded to the first fractional digit. Effectively, this means that r_1 is between 5 and 20 percent of one work cycle. The vehicle travels along straight lines which allow no shortcuts, thus we can assume that the triangle inequality holds, and $r_s = r_{s-1} + rnd(0.05, 0.2)$, $\forall s = 2, \ldots, |S|$. Finally, $r^{SM} = r_{|S|} + rnd(0.05, 0.2)$. Moreover, for each station $s \in S$ and each cycle $t = 1, \ldots, T$, we set $d_{st} = 1$ with a likelihood of 40%. Furthermore, the replenishment time at the supermarket is set to either $P = 3$ cycles (small instances) or $P = 12$ cycles (large instances).

In current industrial practice, that is, at the major German automotive plants that we visited, tow train schedules are almost invariably cyclic, meaning that each station is visited in fixed, unchanging intervals, e.g., every 48 or 96 cycles. However, the tow train may in fact set off from the supermarket more frequently than that; it may simply skip certain stations on certain tours. Therefore, while the route is fixed, the stops along the route may change (albeit also in a set pattern), some stations only being visited on odd tours, some on even tours and others on every tour.

Consequently, we consider a cyclic "default schedule" where the tugger sets off each 12 (small instances) or 48 cycles (large instances). All stations in subset $S^o \subset S$ are visited on odd tours, where S^o is a random subset of S containing 70% of the stations in S. Set S^e denotes the stations visited on even tours. It contains all stations not part of S^o, i.e., $S \setminus S^o$, and a random subset of S^o such that $|S^e| = |S^o| = \lfloor 0.7 \cdot |S| \rfloor$. Effectively, this means that all stations lie on at least one tour, while about 40% lie on both odd and even tours.

Given this "default schedule" and assuming a stopover time of $p = 0.9$, we can exactly calculate at which cycle δ_{sk} the tow train will stop at station s on tour k (if it stops there at all may depend on whether k is even or odd). This allows calculating the exact amount of bins in demand between two consecutive stops at a station. Let ι_{sk} be the number of bins in demand between cycle δ_{sk} and either the cycle when the tugger next stops at the station (which may be either $\delta_{s,k+1}$ or $\delta_{s,k+2}$) or the end of the planning horizon T if δ_{sk} denotes the last stop. If the tugger does not actually stop at station s on tour k, let $\iota_{sk} = 0$. Then the station capacity is set to $c_s = \max_{k=1,\ldots,n}\{\iota_{sk}\}$, and the vehicle capacity is $C = \max_{k=1,\ldots,n}\{\sum_{s \in S} \iota_{sk}\}$. This effectively means that the station and vehicle capacities are such that, using the "default schedule", all deliveries can be made without inter-tour inventory. Further, we assume that the initial inventory at each station is exactly sufficient to cover all demand up until the first arrival of the tow train in the "default schedule".

In this way, we generated 10 small instances and 10 large ones, which are available from the authors upon request.

6.2 Computational results

In order to assess the computational performance of our proposed tabu search (TS) scheme, we pitted it against a default solver, namely CPLEX 12.6.1, tackling the MIP model from Section 5.1. Moreover, we tested several different values for the time limit,

$\theta \in \{10, 60, 300\}$ seconds, and modified the stopover time for each instance, namely $p \in \{0, 0.3, 0.5, 0.7, 0.9\}$.

p	f^*	CPU s. (CPLEX)	θ	gap (TS rnd)	gap (TS)	# opt. (TS rnd)	# opt. (TS)
0	336.4	14.4	10	0.4%	0.0%	8	10
			60	0.0%	0.0%	10	10
			300	0.0%	0.0%	10	10
0.3	399.2	530.4	10	0.0%	0.5%	9	6
			60	0.1%	0.1%	8	8
			300	0.0%	0.0%	9	9
0.5	445.8	429.7	10	0.0%	0.1%	10	8
			60	0.0%	0.0%	10	10
			300	0.0%	0.0%	10	10
0.7	499.6	211.8	10	0.0%	0.0%	10	10
			60	0.0%	0.0%	10	10
			300	0.0%	0.0%	10	10
0.9	516.1	55.0	10	0.0%	0.0%	10	10
			60	0.0%	0.0%	10	10
			300	0.0%	0.0%	10	10

Table 3: Average optimality gaps for the small instances ($|S| = 10$, $T = 24$).

Table 3 shows the optimality gaps for the small instances. Note that CPLEX failed to solve one instance (with $p = 0.3$) to optimality within the time limit of 30 minutes; we excluded this instance from the table. The average solution time (in seconds) is labeled *CPU s. (CPLEX)*, and the average optimal objective value is f^*. θ denotes the time limit for TS in seconds, *gap* is the average relative optimality gap, calculated as $(f^{TS(rnd)} - f^*)/f^*$, and *# opt.* is the number of instances that were solved to optimality by TS. Note that we tested two variants of TS: One, labeled *TS*, is the tabu search exactly as described in Section 5.3, whereas the other, labeled *TS rnd*, is for the most part identical except that it uses only random initial solutions, unlike TS, which also uses the lower bound solution.

The test results clearly indicate that TS is very capable of solving TTSL to (near-)optimality. Even with a time limit of only $\theta = 10$ seconds, the average optimality gap is without exception well below 1% in all cases. The very worst maximum gap measured in one instance is 3.3%. Increasing the time limit smooths away even these few outliers, lowering the optimality gap to 0% across the board for $\theta = 300$ seconds.

As for the large instances, no optimal solutions are available, seeing that in most cases CPLEX cannot solve these problems even to feasibility, even after two hours of computations. The DP procedure from Proposition 4.2, however, can serve as a lower bound for those instances where $p = 0$, relaxing the station capacity constraints. Note that these solutions are not necessarily valid lower bounds for other instances where $p > 0$ because, thanks to the different timing of the stops, solutions with non-zero stopover times may in fact sometimes (though rarely) be better than those with negligible stopover times. For these test problems where $p = 0$, the average gap to the lower bound after $\theta = 300$ seconds is 10.8% for TS rnd and for TS, it is 0%.

To get a meaningful benchmark for all large test problems, we compare the TS results to the cyclic default schedule, where the tow train periodically sets off from the supermarket each 48 cycles, as described in Section 6.1. Note that this type of cyclic schedule is quite common in practice in our experience and can thus be seen as a status-quo reference solution.

Table 4 lists the average default objective value f^{def} as well as the average relative deviation from that value, calculated as $(f^{def} - f^{TS(rnd)})/f^{def}$. Columns *# feas.* contain the number of instances (out of 10) that were solved to feasibility by the heuristics.

The first striking observation here is the remarkable reduction in line-side inventory that can be achieved through optimization as opposed to using simple cyclic schedules: the average decrease over all instances is about 50%. We can expect this value to vary somewhat depending on the exact length of the cycle chosen and other factors, but the trend is clear: cyclic timetables are clearly substantially suboptimal.

It is also remarkable that both TS variants were almost always able to find at the very least a feasible solution, except in a few cases where $p = 0.9$ and the allotted CPU time is short ($\theta = 10$). In all other cases, even with a time limit of only 10 seconds and starting from a completely random initial solution (TS rnd), feasibility could always be established. Keeping in mind that finding a feasible solution is already an NP-complete problem (Proposition 4.1) and that CPLEX failed to do so even after hours of computations, this is anything but self-evident.

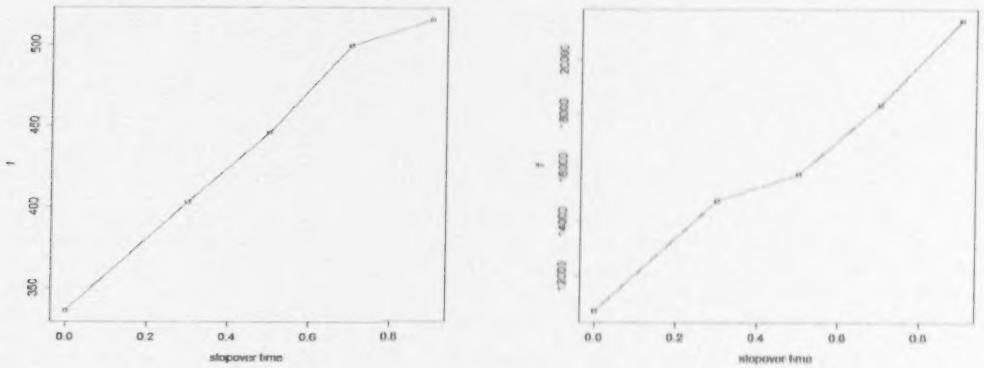
p	f^{def}	θ	dev. (TS rnd)	dev. (TS)	# feas. (TS rnd)	# feas. (TS)
0.0	38833.5	10	65.7%	72.4%	10	10
		60	67.2%	72.4%	10	10
		300	69.4%	72.4%	10	10
0.3	37985.8	10	48.6%	48.9%	10	10
		60	56.2%	57.1%	10	10
		300	60.1%	59.6%	10	10
0.5	37375.8	10	45.2%	42.5%	10	10
		60	56.4%	54.1%	10	10
		300	56.9%	57.1%	10	10
0.7	36761.0	10	40.4%	37.8%	10	10
		60	45.9%	44.7%	10	10
		300	48.4%	49.1%	10	10
0.9	36123.2	10	19.0%	19.2%	9	6
		60	32.4%	33.4%	10	10
		300	36.9%	38.9%	10	10

Table 4: Average performance for the large instances ($|S| = 20$, $T = 144$).

To investigate if the initial solution influences the efficacy of the neighborhood search, we tested for difference of means of the results of TS and TS rnd using a paired two-sample t-test. However, only in the case of $p = 0$ was there a statistically significant difference at the 0.05 level. This may not be surprising since the DP solution used to initialize TS (but not TS rnd) is most probably already very close to optimal for $p = 0$; the only thing left to do for TS is repairing any infeasibilities that are due to violated

station capacities. For $p > 0$, on the other hand, the structure of the optimal timetables is apparently often very different from that of the relaxed timetables, yielding little improvement over random start solutions.

In the final part of our computational study, we will discuss the effect of the stopover time p on work-in-process. Today, many tow train systems in use are still manually handled by a human operator driving the tugger and (un-)loading bins by hand. However, this process is already partially automated to a greater or lesser extent in many assembly plants. For instance, some companies use display panels at each station, indicating with great precision when the tow train is due to arrive next; stopovers can also be partially automated if special gravity flow racks that allow docking are employed. In the most extreme case, the whole process can be fully automated, foregoing human intervention altogether (except in the supermarket), if automated guided vehicles are used. As of today, however, these are rare in existing assembly plants, because they require substantial modifications to racks, driving lanes, and vehicle fleet. This begs the question exactly which level of automation is right. Obviously, a lot of factors influence this decision; in this study, we aim to give some decision support as to the effect of (partially) automated stopovers on in-process inventory.



(a) $|S| = 10$, $T = 24$. (b) $|S| = 20$, $T = 144$.

Figure 4: Effect of stopover time on line-side inventory.

Figure 4 plots the stopover time p against line-side inventory f (averaged over the 10 instances per parameter setting), using the optimal objective values for the small instances and the best heuristic objective values for the large ones. Keeping in mind that the only thing that is varied for each graph is the stopover time (and not the instances as a whole), the impact of even a small increase in stopover time is quite remarkable: going from $p = 0$, i.e., negligible downtimes at the stations as might (approximately) be the case in a fully automated system, to $p = 0.9$, i.e., almost one full work cycle of idle

time per stop as may be appropriate in a completely manual system, about *doubles* total line-side stock. All things considered, p and f seem to be about linearly related, with the other f-values for $0 < p < 0.9$ falling somewhere in between.

Obviously, whether or not investing in new equipment and processes to lower the stopover time is worth the effort (and cost) depends on many other considerations, too. For example, shorter stopovers might also have other benefits, like less congestion owing to the shorter trip time per tour, or savings in personnel cost for fully automated systems. On the other hand, certain types of automation may be hard or even impossible to retrofit on existing shop floors. Either way, our study clearly shows that quite substantial savings with regard to in-process inventory may be achievable, which should certainly be taken into consideration when deciding on the right degree of automation.

7 Conclusion

This paper investigated the problem of scheduling in-plant transport vehicles, commonly tow trains, to feed parts to mixed-model assembly lines, while observing just-in-time objectives. We showed that the problem is NP-complete in the strong sense, although the special case without stopover times and station capacities can be solved in polynomial time. For the general case, we developed a decomposition heuristic, which is shown to produce solutions very close to the optimum.

As for practical advice, we derived the following key points.

- Simple cyclic schedules as they are commonly used in current industrial practice are severely suboptimal, often more than doubling the line-side stock as compared to optimal schedules. Considering that space at the assembly line is very scarce, and avoiding expensive surplus work-in-process is one of the main goals of the just-in-time philosophy, this should be a strong motivation for plant managers to consider using more sophisticated scheduling procedures.
- Decreasing the stopover time, e.g., through automation, can significantly reduce in-process inventory. In fact, inventory and stopover time are about linearly related. Depending on how involved and expensive individual automation measures are, the reduction in line-side stock may well be worth the effort.

Future research should focus on developing specialized exact methods for solving the TTSL. In particular, the decomposition approach proposed in this paper may also form the basis of such an exact procedure. Moreover, our scheduling approach may also be integrated into the problem of determining routes for the tow trains.

References

Arostegui Jr, M.A.; Kadipasaoglu, S.N.; Khumawala, B.M. (2006): An empirical comparison of tabu search, simulated annealing, and genetic algorithms for facilities location problems. International Journal of Production Economics 103, 742-754.

autogramm (2007): Ganz gezielt gesteuert. Internet: http://autogramm.volkswagen.de/12_07/wolfsburg/wolfsburg_06.html, accessed: June 11, 2015.

Battini, D; Faccio, M.; Persona, A.; Sgarbossa, F. (2010): "Supermarket warehouses": stocking policies optimization in an assembly-to-order environment. International Journal of Advanced Manufacturing Technology 50, 775-788.

Battini, D.; Boysen, N.; Emde, S. (2013): Just-in-Time supermarkets for part supply in the automobile industry. Journal of Management Control 24, 209-217.

Blazewicz, J.; Machowiak, M.; Weglarz, J.; Kovalyov, M.Y.; Trystram, D. (2004): Scheduling malleable tasks on parallel processors to minimize the makespan. Annals of Operations Research 129, 65-80.

Boysen, N.; Fliedner, M.; Scholl, A. (2009): Assembly line balancing: Joint precedence graphs under high product variety. IIE Transactions 41, 183-193.

Bozer, Y.A.; McGinnis, L.F. (1992): Kitting versus line stocking: A conceptual framework and a descriptive model. International Journal of Production Economics 28, 1-19.

Choi, W.; Lee,Y. (2002): A dynamic part-feeding system for an automotive assembly line. Computers & Industrial Engineering 43, 123-134.

Emde, S.; Fliedner, M.; Boysen, N. (2012): Optimally loading tow trains for JIT-supply of mixed-model assembly lines. IIE Transactions 44, 121-135.

Emde, S.; Boysen, N. (2012a): Optimally routing and scheduling tow trains for JIT-supply of mixed-model assembly lines. European Journal of Operational Research 217, 287-299.

Emde, S.; Boysen, N. (2012b): Optimally locating in-house logistics areas to facilitate JIT-supply of mixed-model assembly lines. International Journal of Production Economics 135, 393-402.

Fathi, M.; Alvarez, M.J.; Hassani Mehraban, F.; Rodriguez, V. (2014): A Multiobjective Optimization Algorithm to Solve the Part Feeding Problem in Mixed-Model Assembly Lines. Mathematical Problems in Engineering 2014.

Finnsgård, C.; Wänström, C.; Medbo, L.; Neumann, W. P. (2011): Impact of materials exposure on assembly workstation performance. International Journal of Production Research 49, 7253-7274.

Garey, M. R.; Johnson, D. S. (1979): Computers and intractability: A guide to the theory of NP-completeness. W. H. Freeman, San Francisco, CA.

Gendreau, M.; Hertz, A.; Laporte, G. (1994): A tabu search heuristic for the vehicle routing problem. Management Science 40, 1276-1290.

Gendreau, M.; Potvin, J.Y. (2010): Tabu Search. In: Gendreau, M.; Potvin, J.Y. (Eds.): Handbook of Metaheuristics. International Series in Operations Research and Management Science 146. Springer, Berlin, Germany.

Glover, F.; Laguna, M. (1997): Tabu Search. Kluwer, Boston, MA.

Golz, J.; Gujjula, R.; Gunther, H.-O.; Rinderer, S.; Ziegler, M. (2012): Part feeding at high-variant mixed-model assembly lines. Flexible Services and Manufacturing Journal 24, 119-141.

Hertz, A. (1992): Finding a feasible course schedule using tabu search. Discrete Applied Mathematics 35, 255-270.

Kilic, H.S.; Durmusoglu, M.B. (2013): A mathematical model and a heuristic approach for periodic material delivery in lean production environment. The International Journal of Advanced Manufacturing Technology 69, 977-992.

Limère, V.; Van Landeghem, H.; Goetschalckx, M.; Aghezzaf, E.-H.; McGinnis, L.F. (2012): Optimising part feeding in the automotive assembly industry: deciding between kitting and line stocking. International Journal of Production Research 50, 4046-4060.

Orlin, J.B. (1997): A polynomial time primal network simplex algorithm for minimum cost flows. Mathematical Programming 78, 109-129.

Vaidyanathan, B.S.; Matson, J.O.; Miller, D.M.; Matson, J.E. (1999): A capacitated vehicle routing problem for just-in-time delivery. IIE Transactions 31, 1083-1092.